

On the GI-Completeness of a Sorting Networks Isomorphism

Martin Marinov¹, David Gregg²

Abstract

Our main research interest is optimizing n -input sorting networks — a mathematical object oblivious to the order of the input data which always performs the same set of pre-determined operations to produce a sorted list of n numbers. In the early 2000's a sorting/comparator network isomorphism and normalization is presented by Choi and Moon. This is used to substantially reduce the search space for optimal n -input sorting networks by considering only representative networks up to the isomorphism. Choi and Moon prove the computational complexity of checking whether two n -input networks are isomorphic to be polynomially reducible to the bounded valence graph isomorphism (GI) problem. In 2013, Bundala and Zavodny described a new sorting network relation (subitemset isomorphism) which is 'superior' to that of Choi and Moon — any networks that are CM (Choi and Moon) isomorphic are also BZ (Bundala and Zavodny) subitemset isomorphic but the converse is not true in the general case. Bundala and Zavodny's sorting network subitemset isomorphism drastically reduces the search space for optimal sorting networks in comparison to previous methods. Their (BZ) isomorphism is at the core of their computer-assisted proof for depth optimality of n -input sorting networks for $11 \leq n \leq 16$ and also Codish et al's comparator optimality computer-assisted proof for nine and ten-input sorting networks.

The subitemset isomorphism problem is really important and there are excellent practical solutions described in the literature. However, the computational complexity analysis and classification of the BZ subitemset isomorphism problem is currently an open problem. In this paper we prove that checking whether two sorting networks are BZ isomorphic to each other is GI-Complete; the general GI (Graph Isomorphism) problem is known to be in NP and LWPP, but widely believed to be neither P nor NP-Complete; recent research suggests that the problem is in QP. Moreover, we state the BZ sorting network isomorphism problem as a general isomorphism problem on itemsets — because every sorting network is represented by Bundala and Zavodny as an itemset. The complexity classification presented in this paper applies sorting networks, as well as the general itemset isomorphism problem. The main consequence of our work is that currently no polynomial-time algorithm exists for solving the BZ sorting network subitemset isomorphism problem; however the CM sorting network isomorphism problem can be efficiently solved in polynomial time.

Keywords: Itemset Isomorphism, Graph Isomorphism, GI-Complete, GI-Hard, Optimal Sorting Networks

1. Introduction

1.1. Structure

We have structured the presentation of our work in the following manner. First, we give a brief introduction to the sorting network optimization problem to describe one real world instance of the (generic) problem tackled in this paper. Next, we give a formal description of the Subitemset Isomorphism (SI) problem together with necessary terminology. Then, we give a summary of the related work on the complexity classification of the problem. In Section 4, we present our main result by proving the Itemset Isomorphism (II) problem is GI-Complete; an immediate consequence is that the SI problem is GI-Hard. In Appendix A and Appendix B we present a set of examples that illustrate main points of the rather technical complexity classification proofs. We conclude by presenting a brief summary of our work and discuss possibilities for future contributions.

1.2. Terminology

We now precede with the formal definitions all of the mathematical objects that are used throughout this paper. Visual examples of all object types are presented in Figure 1. Unless otherwise stated, we assume to be working in the domain $D = \{d_1, d_2, \dots, d_n\}$ of n distinct elements.

- *item* — a set of elements over the domain D . We represent an item I as a binary string of length n where the i -th bit is equal to 1 iff the element $d_i \in I$ for all $1 \leq i \leq n$; i.e. $I \subseteq \{0, 1\}^n$. See Figure 1(a) for examples of items.
- *itemset* — a set of items over the domain D . We represent an itemset S as a matrix with $|S|$ rows and n columns over the field $\{0, 1\}$. See Figure 1(b) for examples of itemsets.
- *dataset* — an ordered set of itemsets by cardinality in ascending order over the domain D . See Figure 1(c) for examples of datasets.

We have chosen the labels of the objects to match that of itemset mining algorithms [1] [2] [3] [4] because the extremal sets identification problem is a sub-problem of the main task. Codish et al. [5] [6] describe a subset of our problem as ‘words up to permutations’ instead of the generalization of ‘itemsets up to subitemset isomorphism’. We consider the choice of naming objects to be personal preference, because all that is important is the mathematical structure of the object that we work with, not the labels used. Hence, we are as rigorous as possible in this Section 1.2, when it comes to defining the objects.

Email addresses: marinovm@tcd.ie (Martin Marinov), dgregg@cs.tcd.ie (David Gregg)

¹Corresponding author. Dept. of Computer Science, Trinity College Dublin, Ireland. Work supported by the Irish Research Council (IRC).

²Lero, Trinity College Dublin.

$a = \{d_2, d_3, d_4, d_6\} = 0\ 1\ 1\ 1\ 0\ 1\ 0$
 $b = \{d_1, d_4\} = 1\ 0\ 0\ 1\ 0\ 0\ 0$
 $c = \{d_1, d_5\} = 1\ 0\ 0\ 0\ 1\ 0\ 0$
 $d = \{d_1, d_2, d_5, d_7\} = 1\ 1\ 0\ 0\ 1\ 0\ 1$

(a) *Item* — a set of elements over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$. The items a , b , c and d are presented. We can always represent a set over a domain D as a binary string of length $|D|$ where the i -th bit equals one iff the element d_i is contained in the set.

$$S \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline b & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ d & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

$$T \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline a & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ c & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ d & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

(b) *Itemset* — set of items over the domain D . The two itemsets $S = \{b, d\}$ and $T = \{a, c, d\}$ over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ are presented. Remember that there are no duplicate items within an itemset.

$$F = \left\langle S \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline b & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ d & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}, T \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline a & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ c & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ d & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right\rangle$$

(c) *Dataset* — ordered set of itemsets over the domain D . The dataset $F = \langle S, T \rangle$ over the domain $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ is presented. Remember that the itemsets within a dataset are ordered increasingly by cardinality.

Figure 1: Graphical representation of the mathematical objects that are used throughout the paper — item, itemset and dataset. For all of the examples in this figure, we use a dataset $D = \{d_1, d_2, \dots, d_7\}$ of seven elements. For a formal definition, please refer to Section 1.2.

1.3. Operations

Having defined all of the necessary terminology (Section 1.2) we now formally put forward the respective operations that we investigate in this paper.

Definition 1.1. Let S and T be itemsets over the domains D_S and D_T , respectively. We say that S is isomorphic to T iff there exists a bijection $J : D_S \rightarrow D_T$ such that $J(S) = T$, also written as $S \cong T$; where $J(S) = \{\{J(d) \mid d \in I\} \mid I \in S\}$. If $D_S = D_T$ then we refer to J as an automorphism.

Remark 1.2. Let $S = \{S_1, S_2, \dots, S_k\}$ and $T = \{T_1, T_2, \dots, T_k\}$ be itemsets over the domains D_S and D_T such that S is isomorphic to T given by $J(S) = T$. Since S and T are sets there exists a bijection $\sigma : \{1, 2, \dots, k\} \rightarrow \{1, 2, \dots, k\}$ which maps the items from S to the items in T such that $\forall S_i \in S$ we have $J(S_i) = T_{\sigma(i)}$.

Definition 1.3. Let S and T be itemsets over the same domain D . We say that S is subset of T up to isomorphism iff there exists a bijective $J : D \rightarrow D$ such that $J(S) \subseteq T$, also written as $S \preceq T$.

1.4. The Problems of Interest

Definition 1.4. *Itemset Isomorphism (II) decision problem:*

Input: Two itemsets S and T over the domains D_S and D_T , respectively.

Question: Is there a bijection $J : D_S \rightarrow D_T$ s.t. $J(S) = T$?

Definition 1.5. *Subitemset Isomorphism (SI) decision problem:*

Input: Two itemsets S and T over the domain D .

Question: Is there a bijection $J : D \rightarrow D$ s.t. $J(S) \subseteq T$?

What is the worst-case complexity class of any algorithm for solving the II and SI problems?

1.5. Contributions

The main contributions our work can be summarized as follows.

- *Itemset Isomorphism: GI-Complete* — In Section 4 we present a proof that the itemset isomorphism decision problem (equality up to bijection of itemsets) is exactly as difficult as the Graph Isomorphism decision problem. The problem is of great importance in the sorting network optimization domain [7].
- *Subitemset Isomorphism: GI-Hard* — As an immediate consequence, the problem of finding a class representative itemsets up to subitemset isomorphism within a dataset is GI-Hard, that is at least as hard as GI. This problem has been encountered before in recent research [8] [7] [9] [6] [5] in the sorting networks optimization domain, but its worst case computational complexity has never been classified.

2. Motivation: Sorting Network Optimization

2.1. Preliminaries

A sorting network is a mathematical object consisting of exactly n wires and comparators designed to sort an input of n numbers. Sorting networks are oblivious to the order of the input data and always perform the same set of pre-determined operations to produce a sorted list of n numbers. The problem of finding optimal sorting networks was first proposed [10] by Bose and Nelson more than 50 years ago. There are two common measures for the optimality of a sorting network — number of levels (depth) and number of comparators. The problem studied in this paper is central [8] [7] [9] [5] to both sorting network optimization problems.

2.2. The Key Concept

We now restate [7] a key idea in optimizing sorting networks. First, we note that any comparator network can be represented as the itemset of outputs when the network is applied to all possible inputs. The number of possible comparator networks is exponential in the number of channels. Therefore the enormous search space for optimal sorting networks naturally gives rise to the concept of subitemset isomorphism — denoted as \preceq . More specifically, when searching for optimal sorting networks, we can discard comparator networks whose itemset representation is not minimal up to \preceq ; i.e. given a dataset D , we need to find a class representative itemsets up to \preceq within D that are of minimal cardinality. This idea is at the core of recent computer-assisted proves for the level-optimal n -input sorting networks for $11 \leq n \leq 17$ [5] [7] [8] [9] [11]; an algorithm using the same idea was built [6] to provide computer-assisted proof for the comparator-optimal n -input sorting networks for $10 \leq n \leq 11$.

Remark 2.1. *In this Section 2, we omit some important details, due to space and topic limitations, like the exact form/shape of the dataset D where we are allowed to make such search space reduction in the sorting networks optimization domain. For more information on this topic, we refer the reader to the excellent papers [5] [7] [9] [8] by Bundala, Zavodny and Codish et al. They were the first to put forward the idea of subitemset isomorphism (in the sorting networks optimization domain), although not in such a general context as presented in this paper.*

3. Related Work

3.1. Complexity Analysis of CM [12] Sorting Network Isomorphism

Choi and Moon [12] describe a sorting network isomorphism and present an algorithm aimed at reducing the search space in sorting networks optimization. They show that their (CM) isomorphism is polynomial-time equivalent to the Graph Isomorphism problem of bounded valance. The GI problem of bounded valance can be solved efficiently [13] in polynomial time.

The work of Choi and Moon is an inspiration to our work because we examine a stronger (BZ [7]) isomorphism of sorting networks than the CM isomorphism. We prove that the BZ itemset isomorphism problem is polynomial-time equivalent to the (generic) version of the Graph Isomorphism problem. Moreover, we show that the BZ subitemset isomorphism problem is GI-Hard.

3.2. Known Algorithms for the Subitemset Isomorphism Problem

Given a dataset F , the relation \preceq induces a partial order on F . To optimize the search for optimal sorting networks it is enough [6](Section 3) for one to consider only the minimal representative itemsets within F up to \preceq .

Multiple [14] [6] [7], very fast deterministic practical algorithms for finding a class representative of minimal up to \preceq itemsets within a dataset D exist in the literature. However, the worst case complexity of all these known algorithms is exponential in n — the size of the domain/alphabet, as defined in Section 1.2.

In this paper, we narrow the gap between theory and practice by formally proving that the SI decision problem is at least as difficult as the Graph Isomorphism (GI) decision problem. Moreover, we show that the Itemset Isomorphism problem is GI-Complete; that is polynomial-time equivalent to the Graph Isomorphism (GI) problem.

3.3. Complexity Analysis of Graph Isomorphism

The graph isomorphism problem is one of two listed [15] by Garey and Johnson that is yet to be classified. The possible complexity classes include but are not limited to: P, NP-Complete, QP (as very recent research suggests by L. Babai). Over the years, there is substantial research on the GI problem: fast practical algorithms with or without domain restrictions [16] [17] [18], complexity analysis [19] [20] [21] [13], GI-Complete problems [22] [23], etc. More importantly, it is commonly believed that GI-Complete problems form a uniquely defined complexity class that sits between P and NP-Complete, but this is yet to be proved. In this paper, we use the fact that the Hypergraph Isomorphism (HGI) decision problem is polynomial-time equivalent [23] to the Graph Isomorphism (GI) problem.

3.4. Itemset Mining

It is worth noting that the itemset terminology used throughout this paper is widely used in the context of database mining and frequent itemset mining. The graph isomorphism problem correlation to itemset mining is evident in the work of Juan et al. [24] who investigate the subgraph mining problem. Another data mining example is the work of Nuyoshi et al. [25] who use quantitative itemset mining techniques to mine frequent graph patterns.

Lastly, we note that the problem of finding minimal itemsets within a dataset up to subitemset isomorphism is a generalization of the extremal sets problem [14]; where in the extremal sets problem relabelling of domain elements is not permitted. The problem of finding extremal sets [2] [3] [4] has received large attention in recent years. Moreover, practical algorithms [14] for finding minimal itemsets up to subitemset isomorphism within a dataset, first find the minimal itemsets within the dataset to reduce the search space.

4. Complexity Classification

In this section we first formally define the Graph Isomorphism (GI) decision problem [15] [22]. We then precede with an informal discussion of how the GI and II problem “differ”. In Section 4.2 we show that $GI \leq_P II$ and then in Section 4.3 we show that $II \leq_P GI$. Following is a natural deduction that II is GI-Complete and also the natural consequence of the GI-Hardness of SI ($GI \leq_P SI$).

Definition 4.1. *Graph Isomorphism (GI) decision problem:*

Input: Two undirected graphs $G = \langle V_G, E_G \rangle$ and $H = \langle V_H, E_H \rangle$.

Question: Is there a bijection $I : V_G \rightarrow V_H$ s.t. $(v, w) \in E_G$ iff $(I(v), I(w)) \in E_H$?

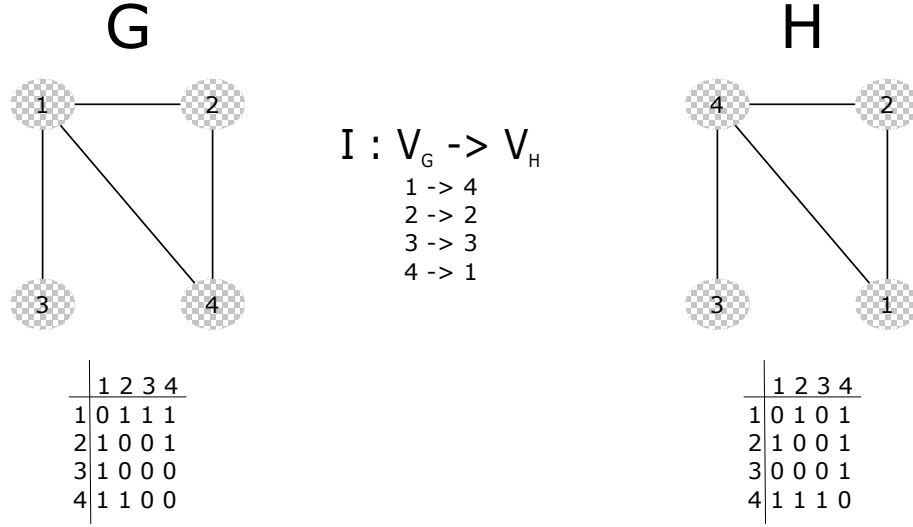


Figure 2: This figure presents a graph G and its adjacency matrix. We also present a swap of the vertices 1 and 4 of G to obtain H . To construct the adjacency matrix of H from the adjacency matrix of G , we first swap the rows 1 and 4 of G and then swap the columns 1 and 4. Since, every permutation (of the vertices of G) can be written as a sequence of swaps, this figure shows the methodology of applying a permutation to a graph.

4.1. Discussion

Before presenting a rather technical proof that the II decision problem is GI-Complete, we give a brief discussion on how the GI and SI problems “differ”. Intuitively, the two problems are very similar as the inputs to both problems can be represented as zero-one matrices — see Figures 2 and 3; however, there are two fundamental differences.

- In the GI problem a swap of vertices is represented as a swap of two rows and two columns of the zero-one adjacency matrix (Figure 2), whereas in the II problem a swap of two domain elements is represented as a swap of two columns (Figure 3) with the rows left intact.
- A valid solution for the GI problem requires the two zero-one adjacency matrices to match exactly. Whereas, in the II problem any reordering of the rows is permitted (recall Remark 1.2).

4.2. II is GI-Hard

Lemma 4.2. $GI \leq_P II$.

The proof of Lemma 4.2 is a rather technical one. However, the proof is constructive, and we present examples in Figures A.4 and A.5 for the essential steps of the proof. A detailed explanation of the examples following the steps of the proof of Lemma 4.2 is presented in Appendix A.

S					J : D -> D					T				
	d ₁	d ₂	d ₃	d ₄							d ₁	d ₂	d ₃	d ₄
1	1	1	1	0		d ₁ -> d ₄					1	0	1	1
2	1	0	0	1		d ₂ -> d ₂					2	1	0	0
3	1	1	0	0		d ₃ -> d ₃					3	0	1	0
4	0	0	1	1		d ₄ -> d ₁					4	1	0	1

Figure 3: This figure presents an itemset S over the domain $D = \{d_1, d_2, d_3, d_4\}$ and its matrix representation (as described in Section 1.2 and Figure 1(b)). We also present a swap of the domain elements d_1 and d_4 of S to obtain T . To construct the matrix representation of T from the matrix of S , we need to swap the two columns d_1 and d_4 . Since, every permutation (of the domain D) can be written as a sequence of swaps, this figure shows the methodology of applying a permutation to an itemset.

Proof. Define the function $f : \langle G, H \rangle = \langle S, T \rangle$ where $\langle G, H \rangle$ is input to GI and $\langle S, T \rangle$ is an input to II. The itemset $S = \{S_u \mid u \in V_G\}$ where the items $S_u = \{(u, v) \in E_G \mid v \in V_G\}$. Similarly, the itemset $T = \{T_h \mid h \in V_H\}$ where the items $T_h = \{(h, w) \in E_H \mid w \in V_H\}$. We now show that the function f is a polynomial-time reduction of Graph-Isomorphism to Itemset-Isomorphism.

First, we need to show that the function f is a polynomial time one. It is obvious, that this is the case, because f does no computation and simply, re-structures the input. Hence, the reduction function f is polynomial time.

To prove that the presented polynomial-time reduction is correct, we need to show that a Graph-Isomorphism instance is satisfiable (yes instance), if and only if the f -induced Itemset-Isomorphism instance is satisfiable.

Suppose that the Graph-Isomorphism instance is satisfiable: there exists a bijection $I : V_G \rightarrow V_H$ s.t. $(v, w) \in E_G$ iff $(I(v), I(w)) \in E_H$. We claim that $J : (v, w) \rightarrow (I(v), I(w))$ satisfies $J(S) = T$. To see this, consider any item $S_g = \{(g, x) \in E_G \mid x \in V_G\} \in S$ and apply the bijection J to it. Then clearly we have $J(S_g) = \{(I(g), I(x)) \in V_H \mid x \in V_G\} = \{(I(g), y) \in V_H \mid y \in V_H\} = T_{I(g)} \in T$. Also note that, since I is bijective then I^{-1} exists and we can similarly show that for any $T_h \in T$ we have $I^{-1}(T_h) = S_{I^{-1}(h)} \in S$. Hence, we have shown that, if any Graph-Isomorphism instance $\langle G, H \rangle$ is satisfiable then the created Itemset-Isomorphism instance $f(\langle G, H \rangle)$ is satisfiable.

Now suppose that the created Itemset-Isomorphism instance $f(\langle G, H \rangle) = \langle S, T \rangle$ is satisfiable: there is a bijection $J : E_G \rightarrow E_H$ s.t. $J(S) = T$. By Definition 1.1 of itemset isomorphism and Remark 1.2, we know there exists a bijection σ that maps the items in $J(S)$ to the items in T . Hence, $\sigma : V \rightarrow H$ is such that for any $S_g \in S$ we have $J(S_g) = T_{\sigma(g)} \in T$, and vice versa. We claim that σ gives a graph isomorphism from G to H . To see this, notice that for all $(v, w) \in E_G$ we have $(\sigma(v), \sigma(w)) = J((v, w))$ (we are working with undirected graphs). But, from the assumption we know that $J((v, w)) \in E_H$; to go from E_H to E_G is systematically the same because σ is bijective, hence σ^{-1} exists. Therefore, we have shown that if the created Itemset-Isomorphism instance $f(\langle G, H \rangle) = \langle S, T \rangle$ is satisfiable then the original Graph-Isomorphism instance $\langle G, H \rangle$ is satisfiable. \square

4.3. GI is II-Hard

In order to prove that there is polynomial-time reduction from the II decision problem to the GI decision problem, we require an intermediate result. Namely, we are going to show that the Hypergraph Isomorphism (HGI, see the following Definition 4.3) decision problem is II-Hard; in other words, we will show that HGI is at least as hard (up to a polynomial transformation) as II. We also use the known fact that HGI is polynomial-time equivalent to GI, formally stated in Theorem 4.4.

We refer to a hypergraph $G = \langle V_G, E_G \rangle$ as a combinatorial object consisting of nodes V_G and edges E_G except that the edges consist of an arbitrary number of nodes within V_G [26]. Examples of hypergraphs are given in Appendix B.

Definition 4.3. *Hypergraph Isomorphism (HGI) decision problem:*

Input: Two undirected hypergraphs $G = \langle V_G, E_G \rangle$ and $H = \langle V_H, E_H \rangle$.

Question: Is there a bijection $I : V_G \rightarrow V_H$ s.t. $A \in E_G$ iff $I(A) \in E_H$?
where $A \subseteq V_G$ and $I(A) = \{I(v) \mid v \in A\}$.

Theorem 4.4. *HGI is GI-Complete.*

Proof. The statement of Theorem 4.4 is very well known in the community [27]. We refer the reader to [23]. \square

To summarize our overall strategy, we prove that $\text{II} \leq_P \text{HGI}$ but since HGI is GI-Complete, we deduce that GI is II-Hard.

Lemma 4.5. *$\text{II} \leq_P \text{HGI}$.*

The proof of Lemma 4.5 is very similar to the proof of Lemma 4.2 from Section 4.2. However, there is a complication: namely, given an instance $\langle S, T \rangle$ of the II problem such that either S or T contain a column with total number of ones not equal to two, in the matrix representation (as described in Section 1.2 and Figure 1(b)). This situation requires a slightly different mechanism to the one described in the proof of Lemma 4.2.

In the following proof of Lemma 4.5 we define a polynomial-time reduction function $g : \langle S, T \rangle \rightarrow \langle G, H \rangle$ from the II problem to the HGI problem. The natural extension of our machinery (proof of Lemma 4.2) to solve the complication is to think of the ones in any column of the matrix representation of an itemset as defining a hyperedge in a hypergraph. As an example, consider the itemset S in Figure 3; then the g -corresponding hyperedge for the column d_1 would be the set of nodes $\{1, 2, 3\}$, and the hyperedge for the column d_2 would be the set of nodes $\{1, 3\}$ — this is a normal edge only because the cardinality of the hyperedge is 2. We give examples of the following constructive proof in Appendix B.

Proof. Let us start by formally defining the function $g : \langle S, T \rangle \rightarrow \langle G, H \rangle$ over the itemsets $S = \{S_g\}$ and $T = \{T_h\}$ over the domains $D_S = \{s_1, s_2, \dots, s_n\}$ and $D_T = \{t_1, t_2, \dots, t_n\}$ respectively. We set $G = \langle V_G, E_G \rangle$ and $H = \langle V_H, E_H \rangle$ s.t. $V_G = \{g \mid S_g \in S\}$,

$E_s = \{g \mid s \in S_g : S_g \in S\}$, $E_G = \{E_s \mid s \in D_S\}$ and $V_H = \{h \mid T_h \in T\}$, $E_t = \{h \mid t \in T_h : T_h \in T\}$, $E_H = \{E_t \mid t \in D_T\}$. We claim that g is a polynomial-time reduction of II to HGI.

It is clear that g can be implemented in a polynomial time, since g performs no calculation and transforms the itemsets S and T to the hypergraphs G and H respectively; where $|V_G| = |S|$, $|E_G| = |D_S|$, $|V_H| = |T|$ and $|E_H| = |D_T|$.

It is now left to show that the Itemset Isomorphism instance $\langle S, T \rangle$ is satisfiable (yes instance) if and only if the g -induced Hypergraph Isomorphism instance $\langle G, H \rangle$ is satisfiable.

Suppose that the Itemset Isomorphism instance $\langle S, T \rangle$ is satisfiable: there exists a bijection $J : D_S \rightarrow D_T$ s.t. $J(S) = T$. Using Definition 1.1, Remark 1.2 and the construction of g , we deduce there exists a bijection $\sigma : V_G \rightarrow V_H$ s.t. $S_g \in S$ if and only if $J(S_g) = T_{\sigma(g)} \in T$. We claim that the bijection σ gives a hypergraph isomorphism between G and H . To prove our claim, consider any $E_s = \{g \mid s \in S_g : S_g \in S\} \in E_G$. However, applying the bijections σ and J to the set of vertices E_s (hyperedge) is equivalent to $\{\sigma(g) \mid J(s) \in J(S_g) : J(S_g) \in J(S)\}$; simplifying (by setting $h = \sigma(g)$ and $J(S_g) = T_h$) gives us $\{h \mid t \in T_h : T_h \in T\} = E_{J(s)} = E_h$. Since, σ and J are bijective, we deduce that σ gives a hypergraph isomorphism between G and H . Hence, we have shown that if the Itemset Isomorphism instance $\langle S, T \rangle$ is satisfiable then the Hypergraph Isomorphism instance $g(\langle S, T \rangle) = \langle G, H \rangle$ is also satisfiable.

Suppose that the Hypergraph Isomorphism instance $g(\langle S, T \rangle) = \langle G, H \rangle$ is satisfiable: there exists a bijection $I : V_G \rightarrow V_H$ s.t. $E_s \in E_G$ if and only if $I(E_s) \in E_H$. However, since $E_H = \{E_t \mid t \in D_T\}$ then we can define a bijection $\gamma : D_S \rightarrow D_T$ such that $s \mapsto t$ if and only if $I(E_s) = E_t$. We claim that γ gives an Itemset Isomorphism between S and T . This is trivially seen because $\gamma(S_g) = T_{I(g)}$ for all $g \in V_G$ and due to γ and I being bijections. \square

Lemma 4.6. $II \leq_P GI$.

Proof. Immediate consequence of applying Theorem 4.4 and Lemma 4.5. \square

4.4. II is GI-Complete

To complete our proof that II is GI-Complete, we still need to show that $II \in NP$.

Lemma 4.7. $II \in NP$.

Proof. We need to show that a polynomial time verifier of the Itemset-Isomorphism problem exists to conclude that II is in NP. It is easy to see that, given a bijection J the verifier needs to check if $J(S) = T$. Clearly the application of the bijection J to S can be done in polynomial time. The equality checking can be done in polynomial time because $J(S)$ and T are sets of a polynomial number of elements each. \square

Theorem 4.8. II is GI-Complete.

Proof. Follows immediately by applying Lemmas 4.2, 4.6 and 4.7. \square

Corollary 4.9. SI is GI-Hard.

Proof. An immediate consequence to Theorem 4.8 because obviously $\text{II} \leq_P \text{SI}$ and $\text{SI} \in \text{NP}$. \square

Furthermore, we deduce that finding a class representative [7] [9] [6] up to subitemset isomorphism is GI-Hard — clearly polynomial-time reducible to the SI problem.

5. Conclusion and Future Work

Fast algorithms for the Subitemset Isomorphism (SI) problem are of practical importance in the sorting networks optimization domain. The SI problem is encountered in recent [8] [9] [7] [6] breakthrough sorting networks optimization research however its worst-case computational complexity classification is an open problem. This current paper proves the Itemset Isomorphism (II) decision problem to be GI-Complete; polynomial-time equivalent to the Graph Isomorphism (GI) decision problem. As a corollary, the SI problem is shown to be GI-Hard. The complexity analysis presented here is of importance to research aimed at fast practical algorithms [6] [14] for the SI problem, as well as, extending the list [22] of GI-Complete problems which are of practical importance too.

For future work, we aim to classify the SI problem more precisely, rather than the lower complexity bound given here. We suspect, that the problem of Subitemset Isomorphism is NP-Complete. The reason, there is an intuitive relation between the pair of Graph-Isomorphism (GI-Complete) and the Subgraph-Isomorphism (NP-Complete) and the pair of Itemset-Isomorphism (GI-Complete) and Subitemset-Isomorphism (GI-Hard).

6. Acknowledgements

Work supported by the Irish Research Council (IRC) and Science Foundation Ireland grant 12/IA/1381.

References

- [1] P. Pritchard, An old sub-quadratic algorithm for finding extremal sets, *Inf. Process. Lett.* 62 (6) (1997) 329–334.
- [2] R. J. Bayardo, B. Panda, Fast algorithms for finding extremal sets, in: *SDM, SIAM / Omnipress*, 2011, pp. 25–34.
- [3] M. Fort, J. A. Sellarès, N. Valladares, Finding extremal sets on the GPU, *J. Parallel Distrib. Comput.* 74 (1) (2014) 1891–1899. doi:10.1016/j.jpdc.2013.07.004.
URL <http://dx.doi.org/10.1016/j.jpdc.2013.07.004>
- [4] M. Marinov, N. Nash, D. Gregg, Practical algorithms for finding extremal sets, *CoRR abs/1508.01753*.
URL <http://arxiv.org/abs/1508.01753>
- [5] M. Codish, L. Cruz-Filipe, P. Schneider-Kamp, The quest for optimal sorting networks: Efficient generation of two-layer prefixes, *CoRR abs/1404.0948*.
URL <http://arxiv.org/abs/1404.0948>
- [6] M. Codish, L. Cruz-Filipe, M. Frank, P. Schneider-Kamp, Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten), *CoRR abs/1405.5754*.
URL <http://arxiv.org/abs/1405.5754>

- [7] D. Bundala, J. Zavodny, Optimal sorting networks, CoRR abs/1310.6271.
URL <http://arxiv.org/abs/1310.6271>
- [8] D. Bundala, M. Codish, L. Cruz-Filipe, P. Schneider-Kamp, J. Závodný, Optimal-depth sorting networks, CoRR abs/1412.5302.
URL <http://arxiv.org/abs/1412.5302>
- [9] D. Bundala, J. Zavodny, Optimal sorting networks, in: A. H. Dediu, C. Martín-Vide, J. L. Sierra-Rodríguez, B. Truthe (Eds.), *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, Vol. 8370 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 236–247.
- [10] R. C. Bose, R. J. Nelson, A sorting problem, *J. ACM* 9 (2) (1962) 282–296. doi:10.1145/321119.321126.
URL <http://doi.acm.org/10.1145/321119.321126>
- [11] T. Ehlers, M. Miller, New bounds on optimal sorting networks, CoRR abs/1501.06946.
URL <http://arxiv.org/abs/1501.06946>
- [12] S. Choi, B. R. Moon, Isomorphism, normalization, and a genetic algorithm for sorting network optimization, in: W. B. Langdon, E. Cantú-Paz, K. E. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. K. Burke, N. Jonoska (Eds.), *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002, Morgan Kaufmann, 2002*, pp. 327–334.
- [13] E. M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *Journal of Computer and System Sciences* 25 (1) (1982) 42 – 65. doi:[http://dx.doi.org/10.1016/0022-0000\(82\)90009-5](http://dx.doi.org/10.1016/0022-0000(82)90009-5).
URL <http://www.sciencedirect.com/science/article/pii/0022000082900095>
- [14] M. Marinov, D. Gregg, A practical algorithm for finding extremal sets up to permutation, TCD-CS-2015-04, Technical Report, Department of Computer Science, Trinity College Dublin.
- [15] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
- [16] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *J. Algorithms* 11 (4) (1990) 631–643. doi:10.1016/0196-6774(90)90013-5.
URL [http://dx.doi.org/10.1016/0196-6774\(90\)90013-5](http://dx.doi.org/10.1016/0196-6774(90)90013-5)
- [17] H. Gazit, J. H. Reif, A randomized parallel algorithm for planar graph isomorphism, in: *Symposium on Parallelism in Algorithms and Architectures, 1990*, pp. 210–219. doi:10.1145/97444.97687.
URL <http://doi.acm.org/10.1145/97444.97687>
- [18] F. Wagner, S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, Planar graph isomorphism is in log-space, *Electronic Colloquium on Computational Complexity (ECCC)* 16 (2009) 52.
URL <http://eccc.hpi-web.de/report/2009/052>
- [19] J. Köbler, U. Schöning, J. Torán, Graph isomorphism is low for PP, *Computational Complexity* 2 (1992) 301–330. doi:10.1007/BF01200427.
URL <http://dx.doi.org/10.1007/BF01200427>
- [20] S. Toda, Graph isomorphism: Its complexity and algorithms (abstract), in: *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, Vol. 1738 of *Lecture Notes in Computer Science*, Springer, 1999, p. 341.
- [21] D. S. Johnson, The NP-completeness column: an ongoing guide, *Journal of Algorithms* 6 (1985) 434–451.
- [22] K. S. Booth, C. J. Colbourn, Problems polynomially equivalent to graph isomorphism, *Computer Science Department, Univ.*, 1979.
- [23] V. Zemlyachenko, N. Korneenko, R. Tyshkevich, Graph isomorphism problem, *Journal of Soviet Mathematics* 29 (4) (1985) 1426–1481. doi:10.1007/BF02104746.
URL <http://dx.doi.org/10.1007/BF02104746>
- [24] J. Huan, W. Wang, J. Prins, Efficient mining of frequent subgraphs in the presence of isomorphism, in: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, IEEE, 2003, pp. 549–552.
- [25] Y. Miyoshi, T. Ozaki, T. Ohkawa, Frequent pattern discovery from a single graph with quantitative

- itemsets, in: Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on, IEEE, 2009, pp. 527–532.
- [26] E. L. Lawler, Cutsets and partitions of hypergraphs, *Networks* 3 (3) (1973) 275–285. doi:10.1002/net.3230030306.
URL <http://dx.doi.org/10.1002/net.3230030306>
- [27] L. Babai, P. Codenotti, Isomorphism of hypergraphs of low rank in moderately exponential time, in: Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on, IEEE, 2008, pp. 667–676.

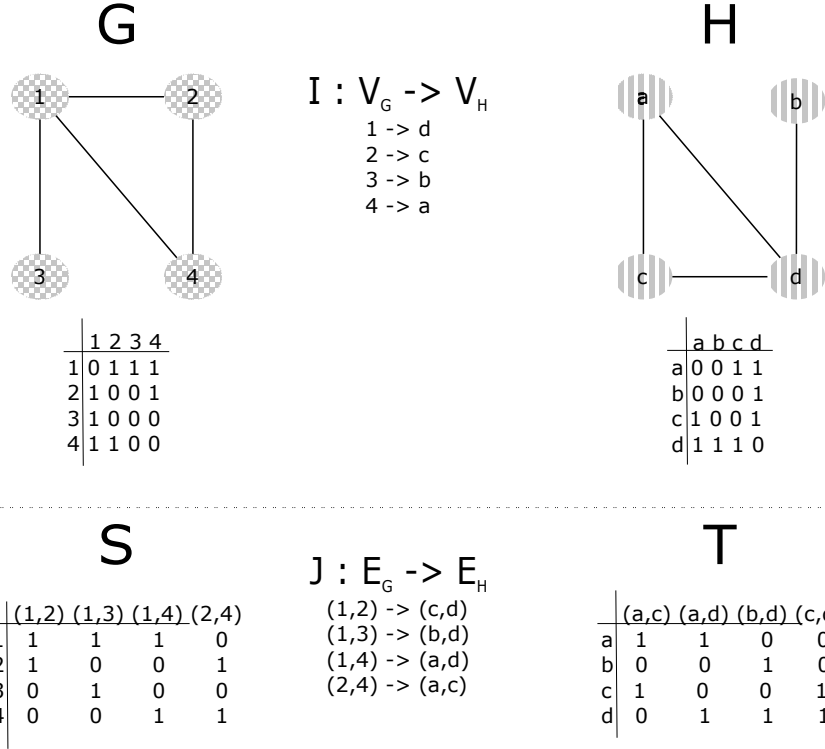


Figure A.4: An example of two isomorphic graphs G and H together with the corresponding isomorphic itemsets S and T generated by the polynomial-time reduction function $f : \langle G, H \rangle = \langle S, T \rangle$, as described in the proof of Lemma 4.2. This figure serves as a detailed example of the constructive proof to Lemma 4.2. In the figure we see that, there is a unique isomorphism between G and H , given by I ; and a unique isomorphism J between S and T . For detailed explanation of this figure refer to Section Appendix A.1.

Appendix A. Examples: II is GI-Hard

The examples presented in Figures A.4 and A.5 demonstrate how to apply the polynomial-time transformation function f (defined in the proof of Lemma 4.2) to an instance $\langle G, H \rangle$ of the GI problem to produce an instance $\langle S, T \rangle$ of the II problem. From the examples, it is clear that $\langle G, H \rangle$ is satisfiable if and only if $\langle S, T \rangle$ is satisfiable.

Following the proof of Lemma 4.2 and the two Figures A.4 and A.5, we see exactly how to construct J using I , and vice versa; where I gives the graph isomorphism between G and H , and J gives the itemset isomorphism between S and T . Note, that Lemma 4.2 works only for undirected graphs; a more technical proof is required for the case of directed graphs but is not necessary for the complexity classification of the itemset isomorphism problem.

Appendix A.1. Figure A.4

The example presented in Figure A.4 shows two isomorphic graphs and their f -corresponding isomorphic itemsets; recall f from proof of Lemma 4.2. It is clear that, the two graphs G and H are uniquely isomorphic — there exists a unique bijection $I : V_G \rightarrow V_H$ that satisfies $(v, w) \in E_G \iff (I(v), I(w)) \in E_H$.

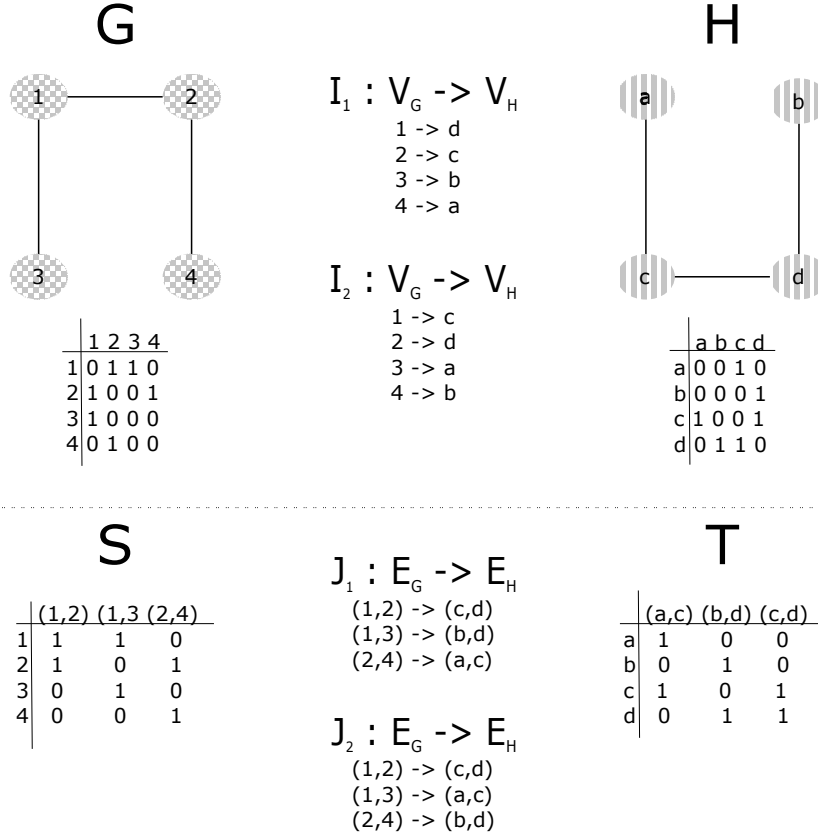


Figure A.5: An example of two isomorphic graphs G and H together with the corresponding isomorphic itemsets S and T generated by the polynomial-time reduction function $f : \langle G, H \rangle = \langle S, T \rangle$, as described in the proof of Lemma 4.2. This figure serves as a detailed example of the constructive proof to Lemma 4.2. In the figure we see that, there are exactly two isomorphisms between G and H , given by I_1 and I_2 ; and exactly two isomorphisms J_1 and J_2 between S and T , where I_1 corresponds to J_1 and I_2 corresponds to J_2 . For an in-depth explanation of this figure refer to Section Appendix A.2.

Hence, given the satisfiable instance $\langle G, H \rangle$ and the bijection $I : V_G \rightarrow V_H$, in the proof of Lemma 4.2 we claim that $J : (v, w) \rightarrow (I(v), I(w))$ satisfies $J(S) = T$. One can easily check the graphs and itemsets in Figure A.4 to verify the correctness of this claim.

Now, suppose we are given a bijection $J : E_G \rightarrow E_H$ s.t. $J(S) = T$. Clearly for the example in Figure A.4, we have a unique $\sigma = I$ that maps the items in $J(S)$ to the items in T .

Appendix A.2. Figure A.5

The example presented in Figure A.5 shows two isomorphic graphs and their f -corresponding isomorphic itemsets. This example is more explanatory than the one presented in Figure A.5 because the isomorphisms between the graphs G and H are not unique. Using the constructive proof of Lemma 4.2, we see that the graph isomorphism I_1 corresponds to the itemset isomorphism J_1 and similarly, I_2 corresponds to J_2 .

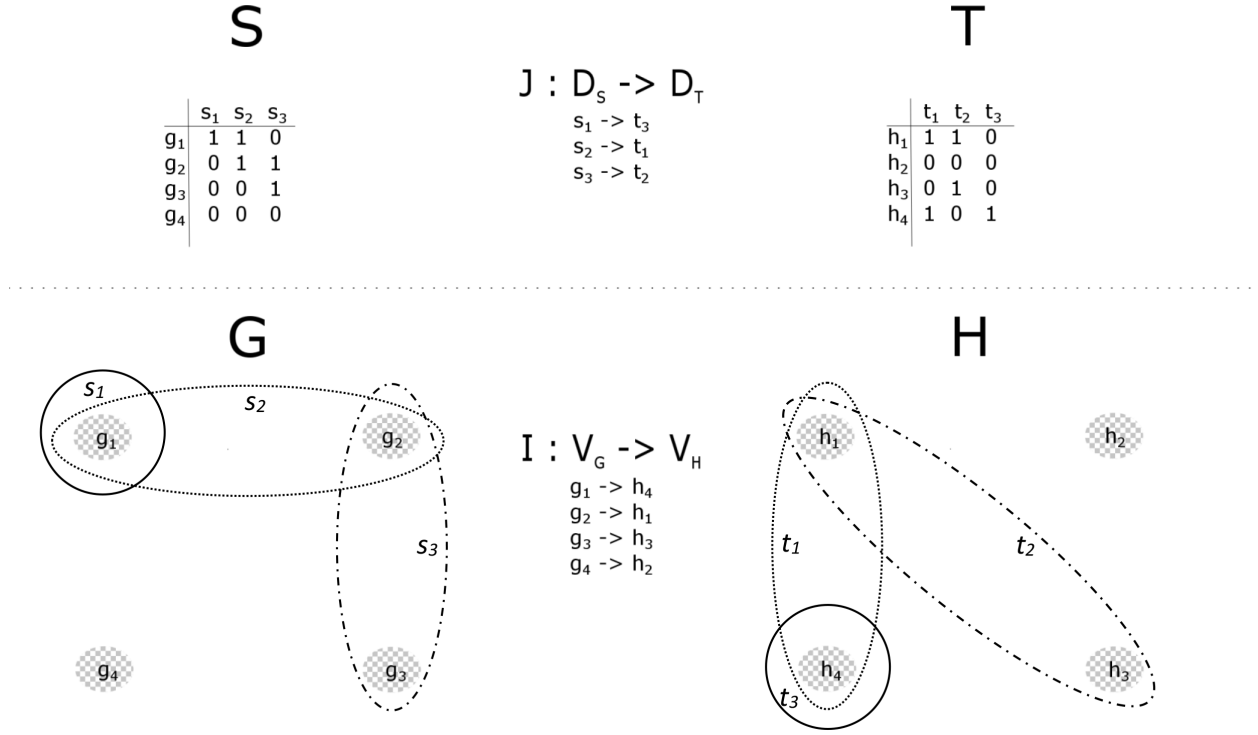


Figure B.6: An example of two isomorphic itemsets S and T together with their g -corresponding isomorphic hypergraphs G and H generated by the polynomial-time reduction function $g : \langle S, T \rangle = \langle G, H \rangle$, as described in the proof of Lemma 4.5. This figure serves as a detailed example of the constructive proof to Lemma 4.5. In the figure we see that, there is a unique isomorphism between S and T , given by J ; and a unique isomorphism I between G and H . For detailed explanation of this figure refer to Section Appendix B.1.

Appendix B. Examples: HGI is II-Hard

The examples presented in Figures B.6 and B.7 demonstrate how to apply the polynomial-time transformation function g (defined in the proof of Lemma 4.6) to an instance $\langle S, T \rangle$ of the SI decision problem to produce an instance $\langle G, H \rangle$ of the GI decision problem. From the examples, it is clear that $\langle S, T \rangle$ is satisfiable if and only if $\langle G, H \rangle$ is satisfiable. Note that we focus our attention on itemsets S and T having at least one column with total number of 1's not equal to two (a proper hyperedge) in their matrix representation (see Section 1.2).

Appendix B.1. Figure B.6

The purpose of this figure is to gently introduce the reader into itemset and hypergraph representation, as well as to cover the basic idea of the proof of Lemma 4.5. Figure B.6 shows two uniquely (given by J) isomorphic itemsets S and T . Following the proof of Lemma 4.5, we claim that $\sigma = I : V_G \rightarrow V_H$ gives a graph isomorphism between the g -induced hypergraphs G and H from S and T respectively. From the figure, it is clear

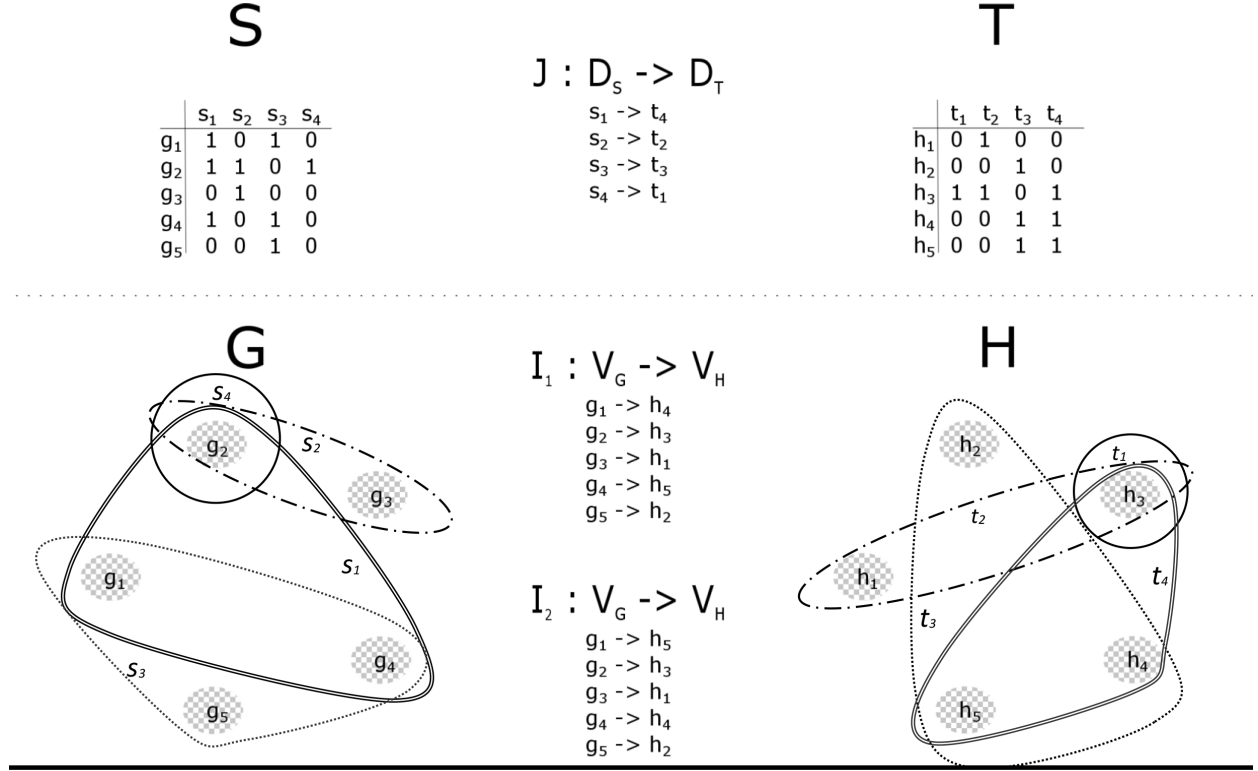


Figure B.7: An example of two isomorphic itemsets S and T together with their g -corresponding isomorphic hypergraphs G and H generated by the polynomial-time reduction function $g : \langle S, T \rangle = \langle G, H \rangle$, as described in the proof of Lemma 4.5. This figure serves as a detailed example of the constructive proof to Lemma 4.5. In the figure we see that, there is a unique isomorphism between S and T , given by J ; and two isomorphisms I_1 and I_2 between the hypergraphs G and H . For detailed explanation of this figure refer to Section Appendix B.2.

that I is a unique isomorphism between G and H ; supporting our claim from the proof of Lemma 4.5 that G and H are isomorphic if and only if S and T are isomorphic, where $g(\langle S, T \rangle) = \langle G, H \rangle$.

Appendix B.2. Figure B.7

Figure B.7 shows a rather complicated scenario of two input itemsets S and T and their g -induced (as defined in the proof of Lemma 4.5) graphs G and H respectively. This case is interesting because there is a unique itemset isomorphism J between S and T , however there are two ($\sigma_1 = I_1$ and $\sigma_2 = I_2$) graph isomorphisms between G and H . Note that the line styles of the hyperedges of both (G and H) graphs are matching if and only if one hyperedge is mapped to the other as given by J ; for example, the hyperedge s_1 in G is mapped to the hyperedge t_4 in H , noting that $J(s_1) = t_4$.